MixedStore: An Enterprise-scale Storage System Combining Solid-state and Hard Disk Drives [*]

Youngjae Kim Aayush Gupta Bhuvan Urgaonkar

Computer Systems Laboratory
Department of Computer Science and Engineering
The Pennsylvania State University, University Park, PA 16802
{youkim, axg354, bhuvan}@cse.psu.edu

Technical Report CSE-08-017
September 2008

ABSTRACT

Flash memory overcomes some key shortcomings of hard disk drives (HDDs), including faster access to non-sequential data (when not degraded by garbage collection (GC) overheads) and lower power consumption. Economic forces, driven by the desire to introduce flash into the enterprise market without changing existing software-base, have resulted in the emergence of solid-state drives (SSDs), flash packaged in HDD form factors and capable of working with device drivers and I/O buses designed for HDDs. Unlike the use of DRAM for caching or buffering, however, certain idiosyncrasies of SSDs make their integration into HDD-based systems non-trivial. Flash memory suffers from limits on its reliability, is an order of magnitude more expensive than the disk, and can be sometimes even slower than the HDD (due to excessive GC induced by high intensity of random writes). Given the complementary properties of HDDs and SSDs in terms of cost, performance, and lifetime, the current consensus among several storage experts is to view SSDs not as a replacement for HDD but rather as a complementary device within the storage hierarchy. We design and evaluate such a hybrid system called *MixedStore* to provide: (a) improved capacity planning techniques to administrators with the overall goal of operating within *cost-budgets* and (b) improved performance/lifetime guarantees during episodes of deviations from expected workloads through three novel mechanisms: *(i) adaptive wear-leveling, (ii) write-regulation and (iii) fragmentation busting*. We implement and validate a simulator for MixedStore and evaluate its efficacy using well-regarded enterprise-scale storage traces. As an illustrative example of MixedStore's efficacy, it is able to reduce the average response time for an enterprise scale random-write dominant Financial Trace by about 71% as compared to a HDD-based system. A preliminary investigation of adaptive wear-leveling allows us to extend the useful lifetime of SSD by about 33% in the presence of unanticipated bursts in I/O, thus opening up new challenges in the design of efficient wear-leveling algorithms.

[*]Name of storage system changed to ensure anonymity.

General Terms

Performance, Experimentation, Measurement

Keywords

Flash Memory, Wear-Leveling, Enterprise-scale Storage systems, Modeling, Performance Prediction, Resource Capacity Planning, Dynamic Data Management

1. INTRODUCTION

Hard disk drives (HDDs) have been the preferred media for data storage in enterprise-scale storage systems for several decades. The disk storage market totals approximately $34 billion annually and is continually on the rise [48]. Manufacturers of HDDs have been successful in ensuring sustained performance improvements while substantially bringing down the price-per-byte. During the past decade, the maximum internal data rate (IDR) for hard disks has witnessed a 20-fold increase resulting from improvements in rotational speeds (RPM) and storage densities; seek times have improved by a factor of 4 over the same period. However, there are several shortcomings inherent to HDDs that are becoming harder to overcome as we move into faster and denser design regimes. First, designers of HDDs are finding it increasingly difficult to further improve the RPM (and hence the IDR) due to problems of dealing with the resulting increase in power consumption and temperature [7, 17, 28]. Second, any further improvement in storage density—another way to increase the IDR—is increasingly harder to achieve and requires significant technological breakthroughs such as perpendicular recording [44, 34, 8]. Third, and perhaps most serious, despite a variety of techniques employing caching, pre-fetching, scheduling, write-buffering, and those based on improving parallelism via replication (e.g., RAID), the mechanical movement involved in the operation of HDDs can severely limit the performance that hard disk based systems are able to offer to workloads with significant randomness and/or lack of locality. Specific to our interest in this paper, in an enterprise-scale system, *consolidation* (e.g., as proposed/explored in [14]) can result in the multiplexing of unrelated workloads imparting/exaggerating the randomness. Furthermore, such consolidated workloads are likely to exhibit degraded temporal and (more seri-

Table 1: Performance, lifetime, cost comparison among different storage media.

Media	Access Time (μs)	Lifetime	Cost($/GB)
DRAM	0.9	N/A	125
SSD	(45) Read , (200) Write	10K-1M Erase Cycles	25
HDD	< 5500	MTTF=1.2Mhr	3

Table 2: Specification of the tested storage device.

	MTron SSD	Western Digital HDD
Model	MSP 7000	Raptor X
Flash Type/RPM	SLC	10,000
Capacity	16GB	150GB
Interface	SATA 1.5GB	SATA 1.5GB

ously for HDD-based systems) spatial locality, thereby potentially adversely affecting performance [14, 15].

Alongside improvements in HDD technology, significant advances have also been made in various forms of solid-state memory such as NAND flash [2], magnetic RAM (MRAM) [43], phase-change memory (PRAM) [19], and Ferroelectric RAM (FRAM) [46]. Solid-state memory offers several advantages over hard disks: lower access latencies for random requests, lower power consumption, lack of noise, and higher robustness to vibrations and temperature. In particular, recent improvements in the design and performance of NAND flash memory (simply *flash* henceforth) have resulted in its becoming popular in many embedded and consumer devices. Small form-factor HDDs have already been replaced by flash in some consumer devices like music players, PDAs, digital cameras, etc. Flash has, however, only seen limited success in the enterprise-scale storage market [32]. Although (i) the aforementioned advances in flash technology and (ii) its dropping cost-per-byte [11] had led several storage experts to predict the inevitable demise of HDDs [12], flash has so far not been able to make inroads into the enterprise-scale storage market to the extent expected [32].

Solid-state Drives. Borrowing a few sentences from an excellent paper on this topic by Leventhal [32], *"The brunt of the effort to bring flash to primary storage has taken the form of solid-state disks (SSDs), flash memory packaged in hard-drive form factors and designed to supplant conventional drives. This technique is alluring because it requires no changes to software or other hardware components, but the cost of flash per gigabyte, while falling quickly, is still far more than hard drives. Only a small number of applications have performance needs that justify the expense".* [1] As evidence of this, major storage vendors producing flash-based large-scale storage systems such as RamSan-500 from Texas Memory Systems, Symmetrix DMX-4 from EMC, ioDrive from ioFusion, etc. are catering only a select class of applications such as large database servers rather than the general enterprise storage market.

Table 1 (all values are based on [32]) presents a comparison of the performance, lifetime, and cost of representative HDDs, SSDs, and DRAM used in the enterprise. There are several important implications of how these properties compare with each other. Flash technology possesses a number of idiosyncrasies that have hindered the SSD from replacing HDD in the general enterprise market. First, it is evident that there exists a huge gap between the Cost/GB of HDDs and SSDs. [2] Second, unlike HDD or DRAM, SSDs possess a

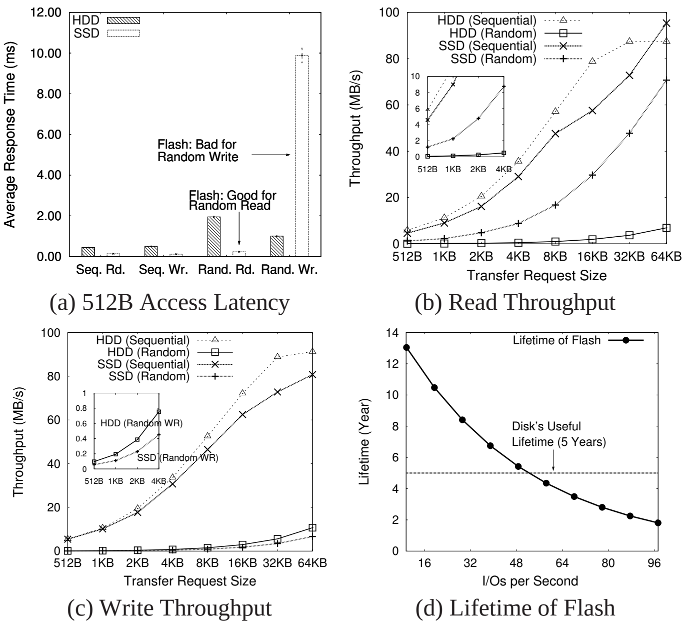


Figure 1: A comparison of the performance and lifetime characteristics of representative SSD and HDD. Although MTTFs for HDDs tend to be of the order of several decades, recent analysis has established that other factors (such as replacement with next, faster generation) implies a much shorter actual lifetime and hence we assume a nominal lifetime of 5 years in the enterprise. Note that Seq., Rand., Wr., and Rd. denote Sequential, Random, Write, and Read. I/O request size in (d) is a page size (2KB). ***Each bar in (a) is shown with 99% confidence interval.***

huge asymmetry between the speeds at which reads and writes may be performed. As a result, the throughput a SSD offers a write-dominant workload is lower than for a read-dominant workload. Third, flash technology restricts the locations on which writes may be performed—a flash location must be *erased* before it can be written—leading to the need for a garbage collector (GC) for/within an SSD. We will elaborate on these properties of flash in Section 2. Certain workload characteristics (in particular, the presence of randomness) increase the fragmentation of data stored in flash memory, i.e., logically consecutive sectors become spread over physically non-consecutive blocks on flash. This exacerbates GC overheads, thereby significantly slowing down the SSD—even to an extent where it operates slower than a HDD! [29]. Furthermore, this slowdown is non-trivial to anticipate. A given set of random writes may themselves experience good throughput, but increase fragmentation, thereby degrading the performance of requests (read or write) arriving much later in future. Finally, to further complicate matters, unlike HDDs, SSDs have a life-time that is limited by the number of erases performed. Therefore, excessive writing to flash, while potentially useful for the overall performance of a flash-based storage system, limits its lifetime. This becomes an important concern in an enterprise-scale employing flash if its workload is write-intensive.

Motivation for MixedStore. From the above description, it should be clear that SSDs are fairly complex devices. Their peculiar prop-

[1]We will use the terms *SSD* and *flash* interchangeably in the rest of this paper.

[2]A similar gap exists between SSD and DRAM. Furthermore, it is projected to worsen in the near future: up to a factor of 13 by 2010 [1]. This rules out major changes in the role played by DRAM in future systems that employ SSDs. DRAM will continue to retain both of its important roles related to caching and buffering. Therefore, we will not compare these two devices in the rest of this paper.

erties related to cost, performance, and lifetime make it difficult for a storage system designer to neatly fit them between HDD and DRAM. To illustrate the complexity of the relationship between HDD and SSD, we perform a simple experiment using the devices described in Table 2. We send raw I/O requests to these actual devices and measure throughput and access latencies. Next, we use our MixedSim simulator (described in detail in Section 5.1) to estimate the useful lifetime of flash-based SSD in MixedStore.

As has been observed in other recent research, under certain workload conditions, an SSD can perform worse than the HDD [29]. A look at Figures 1(a)-(c) provides an illustration of such behavior and calls for careful design to gainfully utilize them in conjunction with HDDs in the enterprise. The degrading lifetime with increased write-intensity, as shown in Figure 1(d), may result in premature replacement of these devices, adding to deployment, procurement, and administrative costs. Note that we have picked a lifetime of 5 years for a HDD just for illustrative purposes. An excellent study of the useful lifetimes of disks based on data from real enterprise-scale systems appears in a paper by Schroeder and Gibson [45]. Finally, the low throughput offered by SSDs to random write-dominated workloads (Figure 1(c)), which are frequently encountered in enterprise-scale systems [29], necessitates intelligent partitioning of data in such hybrid environments while ensuring that the management costs do not overwhelm the performance improvements. As already alluded to and explained in more detail in Section 4, compared to the HDD, an SSD require a longer history to be incorporated into a performance predictor. Modeling these characteristics is an unexplored area and a significant part of our work as well as the foundation of the overall functioning of MixedStore.

Research Contributions. This paper makes the following specific contributions.

- We propose MixedStore, a simplified hybrid storage system containing HDDs and SSDs sharing the I/O bus. Besides this hardware, MixedStore comprises: (i) a *capacity planner* (*MixPlan* henceforth) that makes long-term resource provisioning decisions for the expected workload; it is designed to optimize the cost of equipment that needs to be procured to meet desired performance and lifetime needs for the workload and (ii) a *dynamic controller* (*MixDyn* henceforth) whose goal is to operate the system in desirable performance/lifetime regimes in the face of deviations at short time-scales in workload from those anticipated by MixPlan.
- We develop simple statistical models that MixPlan employs. These models are used in conjunction with MixedSim [3] (a simulator we have developed for MixedStore by enhancing DiskSim [13]) to validate the efficacy of MixPlan for a variety of well-regarded real-world storage traces. We expect MixPlan to provide "rules-of-thumb" to administrators of hybrid storage systems when making provisioning decisions. As an illustrative result, MixPlan is able to identify close to minimum SSD capacity needed to meet a specified performance goal for a realistic random-write dominant workload (Financial Trace [41]).
- We implement MixDyn in our simulator. In a MixedStore prototype, MixDyn would have two components: (a) an enhanced block device driver that employs online statistical performance and lifetime models for SSD (and a performance model for HDD) to dynamically partition incoming workload among the SSD and HDD, and (b) two algorithms within the SSD controller (specifically, within the FTL layer) including reduction of fragmentation within the flash (fragmentation buster) and a novel concept of *adaptive wear-leveling*. As an illustrative result of our empirical evaluation of the efficacy of MixDyn, it is able to prolong the life of SSDs in MixedStore by about 33% in the face of an unexpected increase in I/O activity.
- Finally, we present ideas on how MixPlan and MixDyn could act in concert and present a preliminary validation and evaluation of all components of MixedStore.

[3]Although our simulator is ready for sharing with other researchers, we are unable to provide its URL due to double-blinded review. The name of our simulator has been changed to preserve anonymity. However, reviewers interested in our code and data are welcome to approach us with the permission of the chairs.

Road-map. The rest of this paper is organized as follows. In Section 2, we present the basics of flash technology and discuss relevant related work. Section 2.2 provides a bird's eye-view of the overall MixedStore architecture and how its two components, MixPlan and MixDyn, interact. In Sections 3 and 4, we describe these components and then evaluate them individually as well as when acting together in Section 5. Finally, we present concluding remarks in Section 6.

2. BACKGROUND AND OVERVIEW

2.1 Background on Flash

Basics of Flash Memory Technology. Flash is a unique storage device since unlike the HDD and volatile memories, which provide read and write operations, it also provides an *erase operation* [39]. Salient characteristics of these operations are as follows: Erase operations are performed at the granularity of a *block* which is composed of multiple *pages*. A page is the granularity at which reads and writes are performed. Each page on flash can be in one of three different states: (i) *valid*, (ii) *invalid* and (iii) *free/erased*. When no data has been written to a page, it is in the erased state. A write can be done only to an erased page, changing its state to valid. Erase operations (1.5ms) are significantly slower than reads/writes. Therefore, *out-of-place* writes are performed to existing free pages along with marking the page storing the previous version invalid. Additionally, write latency can be higher than read latency by up to a factor of 4-5. The lifetime of flash memory is limited by the number of erase operations on its cells. Each memory cell typically has a lifetime of 10K-1M erase operations [10]. *Wear-leveling* techniques [23, 25, 33, 6] are used to delay the wear-out of the first flash block. The time-granularity at which wear-leveling is carried out impacts the variance in the lifetime of individual blocks and also the performance of flash: the finer the granularity, the smaller the variance in lifetime.

The Flash Translation Layer (FTL). The FTL is a software layer that translates logical addresses from the file system into physical addresses on flash FTL helps in emulating flash as a normal block device by performing out-of-place updates which in turn helps to hide the erase operation in flash. The mapping table is stored in a small, fast SRAM. These FTLs can be implemented at different granularities of how large an address space a single entry in the mapping table captures. Page-based FTLs map the logical page number of the request sent to the device from the upper layers such as file system to any physical page on flash. However, such translation requires a large mapping table to be stored in SRAM. At the other extreme, in a block-level FTL scheme, only the logical block number is translated into a physical block number whereas the logical page number offset within the block remains fixed, thus reducing the mapping table. However, since a given logical page may now be placed only in a particular physical page within each block, the possibility of finding such a suitable page (at this fixed offset) increases.

To address the shortcomings of the above two extreme mapping

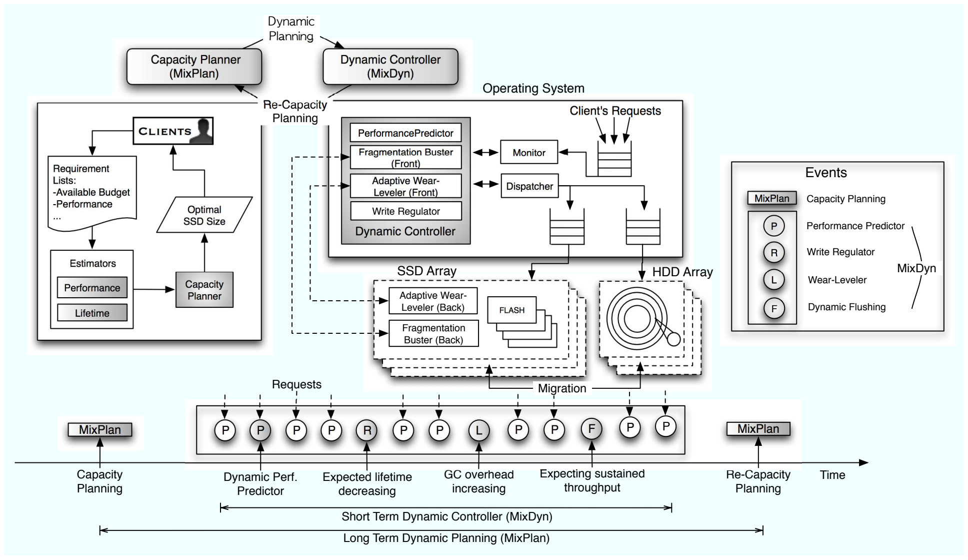


Figure 2: Depiction of various components of MixedStore and how they interact.

schemes, researchers have come up with a variety of alternatives. Although many schemes have been proposed [20, 9, 30, 24, 31], they share one fundamental design principle. Each of these is a *hybrid* between page-level and block-level schemes. They logically partition their blocks into two groups - *Data Blocks* and *Log/Update Blocks*. Data blocks form the majority and are mapped using the block-level mapping scheme whereas the log blocks are mapped using a page-level mapping style. A recent page-based FTL scheme called DFTL [16] utilizes temporal locality in workloads to overcome the shortcomings of the vanilla page-based scheme by storing only a subset of mappings (likely to be accessed) on the limited SRAM and stores the remainder on the flash device itself. We employ DFTL in our evaluation of MixedStore.

2.2 Overview of MixedStore

Figure 2 depicts the interaction between various components of MixedStore. Besides the storage hardware (HDDs, SSDs, and I/O buses) shown in the figure, MixedStore consists of two major software components. The first of these is a long-term resource provisioner called MixPlan. We envision MixPlan to be a tool that would enable storage administrators to provision both kinds of devices in cost-effective ways. The decision-making of MixPlan would occur at coarse time-scales (months to years) corresponding to when procurement and deployment decisions are made. MixPlan employs simple statistical models to make its provisioning decisions, *assuming a priori knowledge* of key characteristics of the workload expected till its next invocation. MixPlan is intended to cost-effectively provision devices to allow MixedStore to (i) adhere to the performance needs of hosted workloads and (ii) meet useful lifetime requirements specified by the administrator, under these workload assumptions. We elaborate on various components of MixPlan in Section 3. The second component of MixedStore is a dynamic controller (MixDyn) that operates at significantly finer time-scales (milliseconds to minutes). MixDyn employs statistical models for performance of SSD and HDD to make dynamic request partitioning decisions—these decisions are made at request-level granularity (milliseconds to seconds). Additionally, it employs novel techniques for data management within the SSD (write regulation, fragmentation busting, and adaptive wear-leveling—all to be elaborated in Section 4) that operate at the granularity of several minutes to hours. Intuitively, the components of MixDyn operate collectively to take corrective data management decisions in MixedStore to adhere to desired performance and lifetime needs despite (i) provisioning errors made by MixPlan and (ii) deviations in workload characteristics and device behavior.

2.3 Related Work

Flash as Cache and Write-Buffer. A lot of research has been conducted to improve performance of HDDs using non-volatile memory. eNVy [49] uses non-volatile memory for data storage wherein battery-backed SRAM is used to reduce the write overhead. MEMS [47] has also been exploited to improve disk performance. Storage architecture in which flash is used as a conventional disk cache has been explored in [36]. Our work goes beyond merely using flash as a cache/write-buffer—rather than treating flash as a *subordinate to the disk*, MixedStore views these as *complementary* storage media. Bisson et al. [5] have explored the use of a flash-based NVRAM as a write buffer to reduce write latency of hard disks for desktop environments. They employ I/O redirection to reduce seeking overhead from disk by directing requests likely to incur long seeks to the on-disk NVRAM. We view the MixDyn component of our system as conceptually close to Bisson et al.'s work and would be interested in comparing MixDyn with their I/O redirection technique in the future. A key difference is that our flash model (developed in Section 3) additionally captures the fragmentation within flash (caused by random writes) and incorporates it into its redirection decision-making. This mechanism will be described in Section 4.

Flash-specific Improvements. Flash Translation Layer (FTL) is one of core-engines in a flash-based SSD. The state-of-the-art FTLs [9, 30, 24, 31] are based on log-buffer based approaches and optimize performance by trying to reduce costly GC overheads. Another orthogonal approach of exposing flash-based devices to the file system has been proposed. JFFS2 [22] and YAFFS2 [50] are the most popular file systems optimized for flash memories. Kim et al. [27] have developed a flash device buffer management scheme to reduce fragmentation caused by random writes. Different SSD designs including interleaving requests to obtain parallelism and ganging etc. have been proposed to improve flash device performance [40]. Further, Managed Flash Technology (MFT) [35] developed by EasyCo is a flash SSD acceleration software which tries to solve flash random write problem by converting random writes into sequential writes at block driver level. Transactional flash (named *TxFlash*) recently proposed by Prabhakaran et al. is a novel SSD that uses flash memory and exports a transactional interface to the higher-level software [42].

Flash in the Enterprise. Kgil et al. [26] have proposed a new architecture named *FlashCache* where they consider replacing a large DRAM with a combination of a smaller DRAM and NAND-based Flash. Their goal is to save memory power consumption while meeting performance requirement by using larger flash and a smaller DRAM. Sun Micro-systems has proposed a storage architecture incorporating flash-based SSDs as intent-log devices and read caches providing improved performance along with reduced power consumption [32]. They propose to use their ZFS file system [51] as an interface to these SSDs. We view Sun's proposed hybrid architecture as the closest in essence to MixedStore and believe that the models and techniques developed here are worth implementing and evaluating in the context of their system. Lee et al., [29] proposed an in-page logging approach in a flash-based DBMS to reduce random write overhead by updating in-place in the database buffer and hence reducing garbage collection overhead. A key contribution in this paper is the observation that workloads with extensive randomness can cause an SSD to perform worse than a HDD. We find similar results in our evaluation and build models that attempt to capture this aspect of an SSD's operation.

Finally, in a recent work from Microsoft Research, Narayanan et al. [38] have also looked at capacity provisioning in hybrid storage systems by utilizing a number of real data center traces available to them. Their work explores the cost-benefit trade-offs of various flash and disk capacities/configurations for these real traces. There are several key differences between our contributions. First, we do not investigate what they call a "two-tiered" hybrid architecture (using SSDs as write buffers/read caches) partly because such ideas have already been explored in the papers cited earlier. Second, we do not explicitly capture power consumption in our formulation of MixPlan (for reasons described in Section 3; indeed as we shall see, our findings are similar to theirs on this front). Third, while they admit that flash wear-out needs to be considered while using it as a write buffer, they do not explore any specific ways of doing this. We incorporate this in the form of lifetime budgets in MixPlan and our dynamic workload partitioning (MixDyn) employs a variety of techniques to adhere to these budgets. Finally, our study goes beyond capacity planning—MixDyn employs a combination of model-driven as well as reactive techniques to operate our hybrid system under given performance/lifetime budgets despite varying workloads. Overall, their work is complementary to MixPlan. Unlike their evaluation with real traces, we are admittedly restricted in our evaluation to publicly-available benchmarks and traces and would greatly benefit from access to their real traces.

3. CAPACITY PLANNING: MIXPLAN

Given the large price gap between SSDs and HDDs, it is useful to be able to determine appropriate capacities of these devices for the workload the system expects to support. We define this process of determining the right size of devices in MixedStore as *capacity planning*. As will be illustrated in Figure 9, both under-provisioning and over-provisioning of flash memory leads to inefficient storage utilization, thus adversely impacting the cost-to-benefit ratio for MixedStore. Therefore, the goal of capacity planning is to minimize this discrepancy so that overall storage investment cost can be optimized.

3.1 Problem Formulation

The objective of capacity planning is to minimize the cost of MixedStore (deployment, management, maintenance etc.) while meeting the service level agreements. These constraints can vary from guaranteeing some minimum performance requirements to reducing management and re-deployment costs, ensuring system reliability etc. For the purpose of our study, we try and minimize the deployment cost (in terms of $) subject to a combination of both performance and re-deployment constraints. We use average system response time as a metric of MixedStore's performance and term this metric as the system's *Performance Budget*. As described in Section 2, the blocks in SSDs become unreliable beyond 10K-1M erase cycles. This poses a significant challenge for a system administrator whose objective is to keep system re-deployment frequency and costs under control. We capture these objectives in terms of a *Lifetime Budget* for the system, which is the time between successive capacity planning decisions and equipment procurement/installation.

We formulate our capacity planning problem as a means of minimizing the cost of acquiring/installing MixedStore while meeting the administrator/workload-specified performance (P_{Budget}) and useful lifetime budget (L_{Budget}). Let C_{SSD} indicate the cost of flash-based SSDs and and C_{HDD} indicate the cost of HDDs in MixedStore. Apart from these, costs associated with power consumption, thermal consumption (cooling), other maintenance and management activity form the recurring costs denoted by C_{Recur}. Then the total MixedStore cost $C_{MixedStore}$ is the sum of these individual costs. Equation 1 shows the formal description of capacity planning. It is easily seen that the above optimization problem reduces to minimizing the cost of SSD for fixed size of HDD available in a MixedStore system. [4]

$$\text{Minimize } C_{MixedStore}$$

$$\text{Subject to } \begin{cases} P_{MixedStore} \geq P_{Budget} \\ L_{MixedStore} \geq L_{Budget} \end{cases} \qquad (1)$$

$$\text{Where } C_{MixedStore} = C_{SSD} + C_{HDD} + C_{Recur}$$

We have shown in another work (reference withheld to maintain anonymity) that the savings in power consumption accrued by utilizing SSDs in enterprise-scale environment are not significant. The same has been corroborated by Narayanan et al. [38] in their recent work. Furthermore, information on how the management/ maintenance costs for HDDs and SSDs compare is still sparse and inconclusive. Hence, we do not consider recurring costs (C_{Recur}) in our current work. The large difference in costs of HDDs and flash-based SSDs (as shown in Table 1) allows us to reduce the capacity planning problem to SSD capacity determination problem for a known sized (constant) HDD-based storage. However, the performance and lifetime of flash-based SSD is highly dependent on not only the workload characteristics but also the internal intricacies of flash such as design of FTL, efficiency of GC etc. This provides a mandate for the design of a robust capacity planner (MixPlan) tool for use by storage system designers. In the next sub-sections, we describe the statistical models utilized by MixPlan to provision SSDs in MixedStore.

3.2 Modeling Performance and Lifetime of Flash Memory for MixPlan

We employ a "black-box" modeling approach for estimating a given SSD's useful lifetime and performance. Our model makes no

[4] A key limitation and difficulty that the reader should note about our capacity planning is that it provisions SSD capacity for the *entire* expected workload rather than the subset of it that is expected to be incident on the SSDs. Clearly, MixPlan over-provisions SSD capacity, potentially heavily. Improving our provisioning on this front is non-trivial and part of ongoing work.

assumptions about the inner configurations (such as FTL employed, SRAM cache size etc.). We do find its efficacy varies depending on the internals of the SSD. For example, the predictor performs better with our page-based like FTL than other state-of-the-art hybrid FTLs. We do not elaborate on these here due to space constraints. For this purpose, we need to identify statistically significant workload characteristics that impact the SSD's lifetime and performance. Performance is directly impacted by data fragmentation caused by random writes which invoke costly GC operations. Moreover, high write intensity increases the number of erase operations required to reclaim invalid space on flash, thus reducing lifetime of blocks. Based on these observations, we consider the following workload characteristics as significant independent variables: (i) average read/write ratio, (ii) spatial locality captured in the form of average sequentiality among requests, (iii) average request inter-arrival time, (iv) average request size, and (v) flash utilization defined as the ratio of the working set size to the total flash size.

3.3 Regression-Based Modeling

Using multiple linear regression, we first find significant predictor variables which affect the variables being predicted: (i) average system response time (ms) for performance budget, (ii) average block erase rate (erases/second) for lifetime budget. The underlying assumption on this linear regression modeling approach is an assumption of linearity. It is assumed that the relationship between variables is linear. Moreover, there is a normality assumption that the residuals follow normal distribution). We start with the general approach in multiple regression of finding significant predictor variables while plugging in as many predictor variables as we can think of. In order to avoid multicollinearity problems, we also perform correlation analysis on predictor variables to ensure that they are all independent variables.

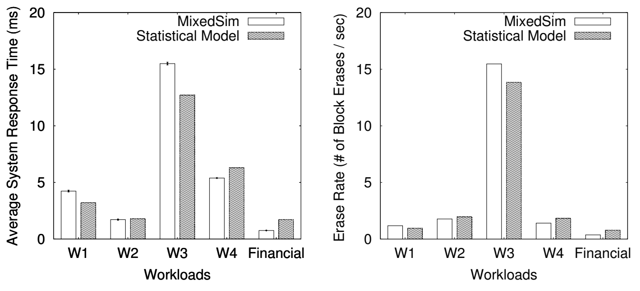
Performance Model for SSD. We use average I/O system response time (R_{avg}) as a predictor of flash performance. I/O system response time represents the time interval between the issuance of request to the SSD by the I/O driver and its completion notification to the driver. It includes queuing delay, bus delay and controller overhead in the device. We first experimented with a multiple linear regression based model. Upon finding this model unsatisfactory, we moved towards a slightly more complicated multiple log-linear model [21]. It can be represented as

$$\log(R_{avg}) = a_0 + \sum_{i=1}^{n} a_i \cdot W_{avg}(i) + \epsilon_1 \qquad (2)$$

where (W_{avg}) represents the average of a particular workload characteristic selected from a set of n parameters discussed earlier (Section 3.2) and ϵ_1 is a small error. The coefficients (a_0, a_1, ... , a_n) are estimated during the learning phase of the experiments.

Lifetime Model for SSD. Erase rate (block erases per second) denoted by E_{avg}, represents the lifetime of a flash device since each block typically has a life of about 10K-1M erase cycles [10]. As in the case of performance modeling, we start by fitting a multiple linear regression model. Again, we observe that a multiple log-linear regression technique, similar to the one used for performance budget is able to model the lifetime budget. The similarity between the two models arises from the fact that higher response times are a function of garbage collection which require block erases and hence impact lifetime. Thus, the lifetime model can be represented as

$$\log(E_{avg}) = b_0 + \sum_{i=1}^{n} b_i \cdot W_{avg}(i) + \epsilon_2 \qquad (3)$$



a) Performance Model (b) Lifetime Model

Figure 3: Validation of performance and lifetime models compared with values measured using MixedSim. ***Each bar of MixedSim in (a) is shown with 99% confidence interval. The 99% confidence intervals of MixedSim in (b) are very small and hence not shown.***

where (W_{avg}) represents the average of a particular workload characteristic selected from a set of n parameters discussed earlier (Section 3.2) and ϵ_2 is a small error. The coefficients (b_0, b_1, ... , b_n) are estimated during the learning phase of the experiments.

3.4 Validation

In this sub-section, we validate our models by comparing against the actual values measured using MixedSim. We generate a large number of synthetic traces by varying workload characteristics described in Section 3.2 to train the models and randomly select 900 of these traces to form our training set. The adjusted R-square [5] is found to be around 90% for both the multiple log-linear models [21]. The average error rate is about 25% for the training set.

Table 3: Some of the synthetic write-only workloads (W1,W2,W3,W4) used to train the performance and lifetime models and a realistic Financial Trace workload used for evaluating the models.

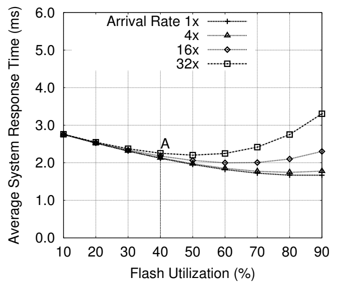
Index	Sequentiality (Ratio)	Request Size (Sectors)	Utilization (Ratio)	Inter-Arrival (ms)
W1	0.10	41.54	0.89	322.18
W2	0.70	16.90	0.89	79.90
W3	0.30	115.71	0.94	80.24
W4	0.70	115.44	0.58	319.74
Financial	0.03	6.57	0.91	164.49

We validate our performance and lifetime models by comparing their results with the corresponding values measured using MixedSim. Table 3.4 shows the salient characteristics of some of the synthetic and real workloads. We choose write-only synthetic traces for validation since flash performs very well for read dominant workloads. Moreover, lifetime is not an issue for such workloads since they encounter very few erase operations. For W2, the error in the performance model is only about 4% whereas it rises to about 21% for W3 which has the highest erase rate and response time values (owing to large request sizes and low inter-arrival times) in the traces shown. For the Financial trace [41], the observed performance as well as lifetime errors are high. The major cause of this discrepancy is that our black-box model assumes no information about the in-

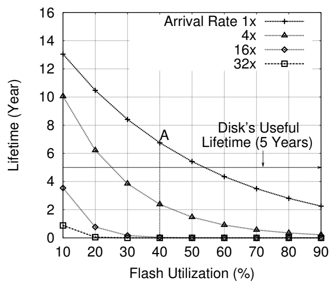
[5]Adjusted R-square defines the proportion of variability that is accounted for by a statistical model. Unlike R-square, it only increases if a newly added predictor statistically improves an existing model.

ternal state of the flash and hence is liable to errors. Arguably, by incorporating more information about flash internals we can improve our model further. However, as explained in Section 2.2, for MixedStore, having a MixPlan suffices so long as MixDyn can handle the inaccuracies in the former models. To summarize our validation, we have demonstrated the possibility of developing a performance and lifetime estimation methodology with reasonable accuracy with simple log linear regression models.

3.5 Why MixPlan Alone Doesn't Suffice



(a) Overall Performance



(b) Lifetime of flash

Figure 4: Capacity planning for Financial-like trace. Note that we increase arrival rate as shown in legends. "A" denotes that the performance and lifetime budgets provisioned by MixPlan. With increased workload arrival rate these guarantees are violated.

Workloads are known to exhibit variation from their predicted behavior. In such circumstances, capacity planning alone is not sufficient to meet the lifetime and performance budgets. Figure 4(a)-(b) show the impact of increased arrival rate on performance and lifetime budgets for a write-dominant workload. If the system designer had provisioned the system at point A to keep the flash lifetime around a disk's useful life while satisfying the performance needs, these guarantees do not hold if the workload changes. With higher intensity of writes, the garbage collector is invoked more often; thus degrading the system's performance. Moreover, it results in higher number of block erases, reducing the flash lifetime. Thus, we require additional sophisticated data partitioning mechanisms which can dynamically adapt to these changing workload environments. In the next section, we describe some techniques employed by our dynamic controller (MixDyn) to meet the various budgets and thus work in synchronization with MixPlan.

4. DYNAMIC CONTROLLER: MIXDYN

We have established the need for a fine-grained control mechanism which should be able to manage the requests and hence ensure sustained throughput from the storage system which is able to meet our lifetime and performance budgets. In this section, we discuss the core of MixDyn—the performance prediction module— and then elaborate other components, namely *(i) Fragmentation Buster*, *(ii) Write Regulator*, and *(iii) Adaptive Wear-leveler* which try to ensure that the guarantees made by MixPlan are upheld.

4.1 Performance Prediction Model for SSD

The performance of the SSD is highly dependent on the workload incident on it. Since out-of-place updates are performed on the flash, GC resulting from fragmentation has an important impact on response time. We build upon our learning from capacity planning and try to develop time-scale performance models suitable for the MixDyn. Although the large-body of work on modeling disk performance is of use here, there are certain salient novel aspects of flash operation that MixDyn's SSD model must capture. Perhaps the most important such feature is that unlike a disk, *an SSD performance model needs to incorporate a much longer history*, since a large enough number of random writes (that might themselves experience good performance) might cause fragmentation over time and the resulting GC invocation would then degrade the performance of requests that arrive much later.

Again we start with identifying the crucial workload characteristics which play a major role. However, contrary to the earlier MixPlan performance model here, we work with a sliding window of requests. This sliding window acts as a short term history of requests and enable us to make fair short term decisions. The main workload characteristics used in the model are: (i) *Average Read to write ratio* of a window of requests, (ii) *Spatial locality*—average sequentiality of a window of requests, (iii) *Request inter-arrival time*, and (iv) *Current request size*. Since this performance model needs to make predictions about the performance of requests in the immediate future, and as seen how performance depends on long-term history, we need to capture and preserve certain aspects of the *current state* of the flash device. However, this information about state of the flash device might require information about SSD internals that may not be feasible (e.g., in the SSD that MixedStore assumes).

In order to build a feasible as well as efficient black-box performance model, we use the history of previous device service times as an indicator of flash device state. For simplicity, we use the average of the service times (S_{avg}). Moreover, we use system response time ($R_{current}$) as a measure of flash device performance. Thus, our multiple linear regression model can be represented as

$$R_{current} = c_0 + c_1 \cdot W_{window} + c_2 \cdot S_{avg} + \epsilon$$
$$S_{avg} = \frac{\left(\sum_{j=1}^{w} S(j)\right)}{w} \qquad (4)$$

where ϵ is a small error and W_{window} is the workload during window w. The coefficients (c_0, c_1, c_2) are estimated during a learning/training phase of our experiment which consists of half of the workload. We believe converting our learning-based prediction technique can be easily adapted to operate on-line, although we do not evaluate that here.

4.2 Evaluation with Dynamism-Aware Performance Prediction Model

Table 4: Statistics for Performance Prediction of Flash for Financial Trace. Correlation between all predictor variables are almost zero.

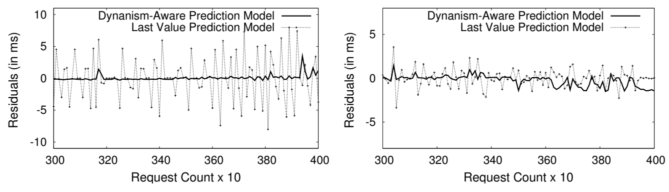
Multiple R	0.98
R Square	0.98
Adjusted R Square	0.98
Standard Error	0.27
Observations	32289

(a) Regression Statistics

	Coefficients	Standard Error	P-value
Intercept	0.13145	0.002982131	0
Previous Device Service Time	0.01223	0.003517166	0.000505959
Real/Write Ratio	-0.66566	0.019041997	8.0937E-263
Sequentiality	-0.78597	0.159642759	8.55189E-07
Inter-Arrival	-0.00007	8.29538E-06	3.00991E-16
Request Size	0.12127	0.000381543	0

(b) Significance of Predictor Variables

We use the Financial trace [41] and TPC-H [52] workload to validate our model. Contrary to our performance predictor for MixPlan, our empirical evaluation suggests a simpler multiple linear regression to be satisfactory. For Financial trace, we observe the measured R-square value to be 98% (as shown in Table 4). We compare the



(a) Financial (b) TPC-H

Figure 5: Comparison of our dynamic SSD performance prediction model with a simple last value-based prediction model. ***The 99% confidence intervals are very small and hence not shown.***

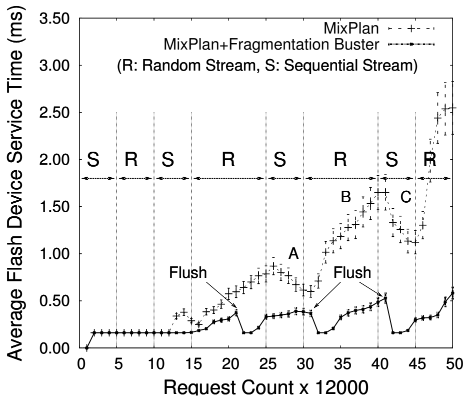


Figure 6: Performance degradation due to fragmentation on flash and subsequent improvement with fragmentation buster. "Flush" indicates periods of migration activity from flash to disk. ***Each point is shown with 95% confidence interval.***

accuracy of our model with a simple baseline—a *last value-based* prediction model for SSD which uses the last service time value as its prediction. Figure 5 demonstrates the superior prediction quality of our model for both TPC-H and Financial trace. Our model is able to predict the state of the flash better than the last value predictor and hence shows much smaller error rate.

4.3 Fragmentation Busting

As described in Section 1, small random writes increase data fragmentation on flash, thus exacerbating garbage collection overhead. We demonstrate this impact in Figure 6 by alternating sequential and small random write requests for synthetic workloads. Both "A" and "C" are regions with sequential write activity. However, the presence of random writes in region "B" leads to data fragmentation on flash, thus increasing the average response time for requests in "C". In order to prevent such fragmented zones on flash, we develop a flushing methodology called *Fragmentation Busting*. As shown in Figure 6, flushing some portion of these small random writes to disk (periodically moving 25% of random writes for this experiment), we can reduce the variation in response times and improve the performance.

Workloads are known to exhibit periods of idleness between bursts of requests [37] providing opportunities for fragmentation busting. Lot of research has gone into developing techniques to identify and utilize these idle periods. Specifically, Mi et al. categorized workloads based on idle periods into tail-based, body-based and body+tail based [37]. and found the presence of heavy-tailed inter-arrival times in enterprise-scale workload implying the presence of significant idle periods along with those of intense activity. Currently, we do not incorporate any specific policy in our MixedStore design. Flushing requires co-operation from the device since the effective mapping tables are present within the device and are not exposed to outer systems. Thus, only a part of the flushing mechanism, specifically the scheduler, can be implemented with MixDyn. In order to decide which data needs to be flushed, the device controller needs to pin the pages causing this fragmentation. We maintain a LRU (Least Recently Used) list of the valid pages using the logical page number of the requests. This represents the cold data on flash and its migration to disk does not have any major impact on MixedStore's performance. When the idle period kicks in, the fragmentation buster directs the flash controller to start flushing the data fragments. A small DRAM-based buffer needs to be maintained so that any request to the data being migrated can be serviced. Since we flush mostly cold data, such requests are rare. Moreover, since this activity can be delayed until an idle period is available, in this work we consider it a pure background activity that does not interfere with the servicing of requests and hence we ignore its possible degrading effects on overall performance.

4.4 Handling Uncertainties in Enterprise-scale Workloads

As described in (Section 3.5), one of the challenges in capacity planning is the unpredictability in workloads. A prolonged and/or recurring period of unanticipated random writes detrimentally impact on lifetime of flash. In this sub-section, we develop techniques for handling sudden unanticipated bursts in requests.

Write Regulation. The projections made by MixPlan are dictated by normal workload characteristics and are subject to violations during operation. The write regulator monitors the erase rate of blocks and comes into action if sustained violations (due to unanticipated write activity) are observed. This is essential to preserve the lifetime budget requirements. On detecting violations, it starts to regulate the writes being sent to flash by over-riding the decisions made by the performance model in MixDyn. Currently, we use a policy which randomly picks the requests being sent to flash and diverts them to disk instead. As part of future work, we plan to develop more sophisticated models.

Adaptive Wear-Leveling. As described in Section 2, wear-leveling requires swapping of data between blocks which have high erase count with blocks which have relatively lower erase count. This swapping operation results in additional erase operations which reduce the lifetime of blocks [6, 33]. These extra erases start to play a significant role towards the end of a flash device's life and indeed accelerate its death. Traditional wear-leveling algorithms define the lifetime of flash based on the reduction in the capacity of the device as compared with the original capacity and hence aim to achieve uniform distribution of erases across all blocks on flash. We propose a paradigm shift in this philosophy by defining the useful lifetime of flash in hybrid environment to be the time till which MixedStore is meeting the performance/lifetime guarantees. This provides us the flexibility to allow wear-out of few blocks on flash by temporarily halting wear-leveling mechanism if it helps in meeting the overall lifetime budget. We propose an *adaptive wear-leveling mechanism*—a novel idea to the best of our knowledge—which, like the write regulator monitors the erase rate of blocks and during periods of prolonged unanticipated write activity, co-ordinates with the flash controller to prevent the extra erases caused by wear-leveling by temporarily halting the leveling algorithm. Once normal I/O activity starts (as projected by MixPlan to uphold the lifetime budget), it allows the device to revert to its wear-leveling mechanism.

5. EVALUATION

5.1 Experimental Setup and Workloads

Workloads. Table 5 illustrates the characteristics of enterprise-scale workloads used in our evaluation. We employ a write-dominant I/O trace from an OLTP application running at a financial institution [41] made available by the Storage Performance Council (SPC), henceforth referred to as the *Financial trace*. We also experiment using Cello99 [18], which is a disk access trace collected from a time-sharing server (exhibiting significant writes) which was running the HP-UX operating system at HP labs. TPC-H [52] is is an ad-hoc, decision-support read dominant benchmark (OLAP workload) examining large volumes of data to execute complex database queries. Finally, we also use a number of synthetic traces to study the efficacy of MixPlan and MixDyn for a wider range of workload characteristics than those exhibited by the above real-world traces.

Table 5: Enterprise-Scale Workload Characteristics. All values are presented as average. Sequentiality refers to requests to logically consecutive addresses.

Workloads	Request Size (KB)	Read (%)	Sequentiality (%)	Inter-arrival Time (ms)
Financial (OLTP) [41]	4.38	9.0	2.0	133.50
Cello99 [18]	5.03	35.0	1.0	41.01
TPC-H (OLAP) [52]	12.82	95.0	18.0	155.56

Table 6: Default simulation parameters.

Flash Device	
Parameter	Value
Flash Type	Large Block
Page (Data)	2KB
Page (OOB)	64B
Block	(128KB+4KB)
Page Read Time	130.9 us
Page Write Time	405.9 us
Block Erase Time	1.5 ms
Interface	SATA
Garbage Collector	Yes
Wear-leveling	Implicit/Explicit
FTL Type	Page/DFTL

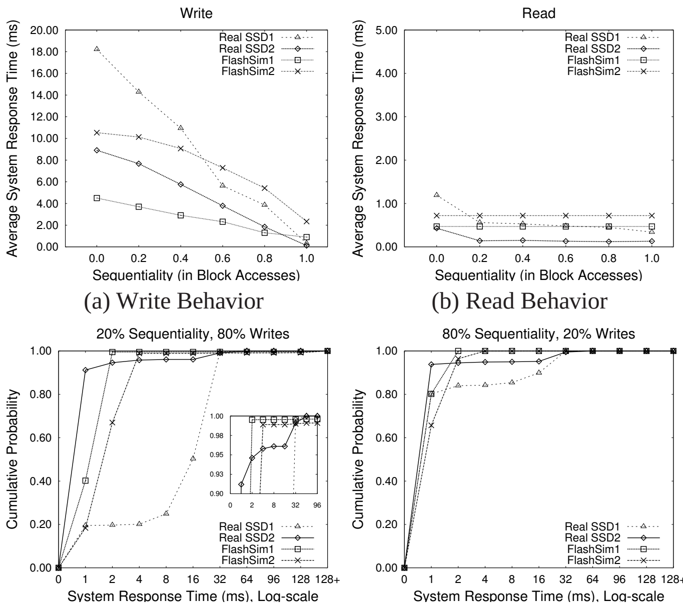
Hard Disk Drive (HDD)	
Parameter	Value
Disk Model	IBM Ultrastar 36Z15
Interface	SATA
Storage Capacity	36.7 GB
RPM	15,000
Seek Time	3.4 msec
Rotation Time	2 msec
Internal Tx Rate	55 MB/sec

MixedSim. We develop a simulation framework for integrated disk and flash based storage systems, called MixedSim. It is built by enhancing Disksim 3.0 [13], a well-regarded HDD simulator. MixedSim is designed with a modular architecture with the capability to model a holistic storage environment. It is able to simulate different storage sub-system components including device drivers, controllers, caches, flash devices, disks, and various interconnects. In our integrated simulator, we add the basic infrastructure required for implementing the internal operations (page read, page write, block erase etc.) of a flash-based device. The core FTL engine is implemented to provide virtual-to-physical address translations along with a garbage collection mechanism.

Note on Our Evaluation Technique. MixedSim is capable of simulating multiple HDDs and SSDs. However, for our evaluation we consider a simple system consisting of a single HDD and SSD with the parameters described in Table 6. [6] We understand that our evaluation does not capture benefits/concerns related to parallelism and fault-tolerance that a system with multiple devices offers. We view our current evaluation as a first step towards understanding performance/cost/lifetime tradeoffs in MixedStore and hope to use the lessons learnt here to expand our evaluation and understanding of hybrid systems.

[6]Simulations using current state-of-the-art HDDs such as Seagate's Cheetah15K and SSDs such as Intel's X25-M SSD are part of our future work.

5.2 Validation of SSD Simulator



(a) Write Behavior (b) Read Behavior

(c) Random Write Dominant (d) Sequential Read Dominant

Figure 7: Validation of our SSD Simulator. Note that in the legends, Real SSD1, Real SSD2, FlashSim1, and FlashSim2 denote Mtron's SSD, SuperTalent's SSD, a SSD using a page-based FTL, and a SSD using DFTL. ***The 99% confidence intervals are very small and hence not shown.***

The specifications available for commercial SSDs are insufficient to model them accurately. For example, the SRAM cache size for FTL mappings, the exact FTL scheme used, etc. are not disclosed. Hence, it is difficult to simulate these commercial devices and we make suitable assumptions for flash device as described in Table 6. Using these parameters, we validate our flash device simulator (a part of MixedSim) against commercial SSDs (MTron's SSD [3] and Super-Talent's SSD [4]) for *behavioral similarity*. For this purpose, we send raw I/O requests to real SSDs and similar traces to our flash device simulator to measure device performance. As shown in Figure 7, our simulator is able to capture the performance trend exhibited by the real SSDs. With increasing sequentiality of writes (Figure 7-(a)), the performance of real SSDs improves and MixedSim with different FTLs is able to provide similar characteristics. In case of reads (Figure 7-(b)), real SSDs show much less variation, the same is observed with our simulator. With high degree of randomness in writes (80% random in Figure 7-(c)) real SSDs demonstrate long-tailed response time distribution (due to larger GC overhead) and our simulator exhibits similar trend. We use page-based FTL along with other flash parameters for the rest of our evaluations. Furthermore, by incorporating more knowledge about SSD internals, we can further improve the validation of our flash simulator and this is part of our future work.

5.3 Evaluation of MixPlan

5.3.1 Lifetime Budget Constraint

Table 7 shows the flash device lifetime for various capacity planning techniques with dynamism-aware data partitioning policy for

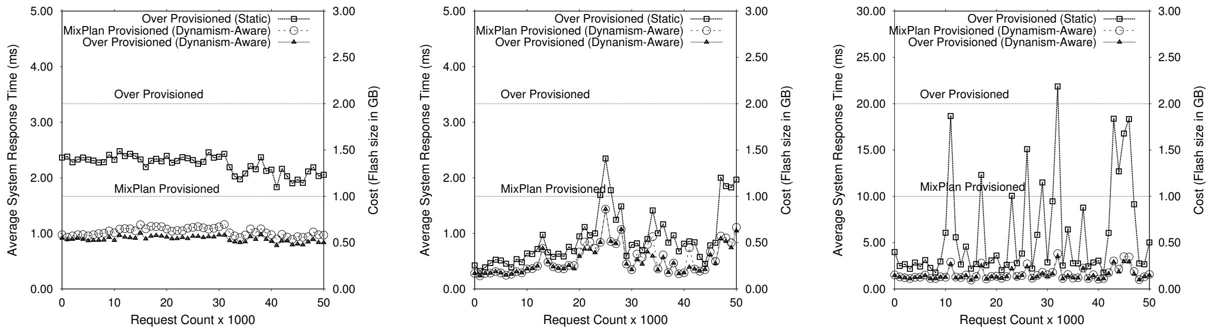


Figure 8: Capacity Planning: MixPlan is not only able to reduce the cost but also improve performance in conjunction with dynamism aware data partitioning. "Static" denotes a static data-partitioning policy where write requests larger than 4KB are assumed to be sequential and are serviced by the HDD and others are serviced by SSD.

Table 7: Lifetime observations with different approaches. A block is assumed to possess 10K reliable erase cycles.

Workload	Lifetime (Yr)		
	Over Provisioned (2GB)	MixPlan (1GB)	Under Provisioned (0.5GB)
Financial	52.69	7.29	2.67
TPC-H	97.50	21.65	-

different workloads. TPC-H is read dominant and hence performance budget is of greater concern than lifetime. For Financial trace which is a write-dominant workload, we observe that under-provisioning capacity would necessitate flash device replacement within 3 years and hence would impact the overall lifetime budget of MixedStore. We want the flash device to last till around the useful life of disk (approximately 5 years) and both over-provisioning and MixPlan are able to achieve this mandate. Over-provisioning flash capacity should reduce the request response times from flash device since the garbage collection overheads will be reduced and hence improve MixedStore performance as compared to MixPlan. However, as we observe in the next sub-section, benefits accrued with this extra flash are much less as compared to the increased cost due to larger flash.

5.3.2 Performance Budget Constraint

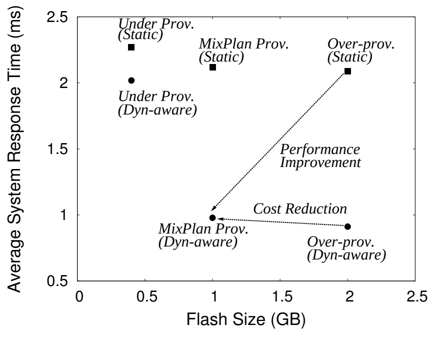


Figure 9: Capacity Planning for Financial Trace. "Static" denotes a static data-partitioning policy where write requests larger than 4KB are assumed to be sequential and are serviced by the HDD and others are serviced by SSD. "Dyn. aware" denotes an intelligent data partitioning.

Figure 9 demonstrates the improvement in performance and reduction in cost using MixPlan along with dynamism-aware performance predictor for Financial Trace. Both MixPlan and Over-provisioning with static data partitioning are able to meet the lifetime guarantees and improve the response time as compared to an under-provisioned system. However, as illustrated in Figure 8(a), if dynamism-aware data partitioner is utilized along with an over-provisioned flash, we observe a slight improvement in performance as compared to MixPlan. But this small improvement comes at an additional cost of bigger flash memory. Thus, the cost-to-benefit (Figure 9) ratio advocates the use of MixPlan for capacity planning in enterprise-scale systems.

As shown in Figure 8(b), for read-dominant TPC-H [52], both MixPlan and over-provisioned models provide similar performance. This can be directly attributed to the fact that read-oriented workloads have very small amount of writes, thus the garbage collector is invoked very infrequently and the service patterns remain similar for both the capacity planning methodologies. Figure 8(c) clearly illustrates the need for dynamism-aware data partitioner (MixDyn). Static partitioning is unable to handle periods of bursts in Cello resulting in poor response times. On the other hand, MixDyn is able to respond effectively to such situations and hence provide better performance. Now we evaluate these benefits in the next sub-sections.

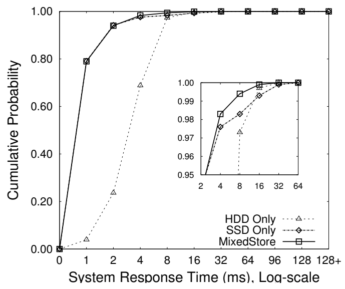
5.4 MixDyn Acting in Concert with MixPlan

We evaluate the performance of prediction models in MixDyn along with our novel three mechanisms such as (i) adaptive wear-leveling, (ii) write-regulation and (iii) fragmentation busting.

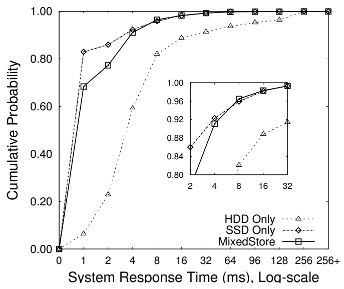
5.4.1 Dynamism-Aware Performance Prediction

We integrate our SSD prediction model with an admittedly simple disk performance predictor. We use a model based on the average response time observed during the training phase to predict disk performance. The dynamic controller (MixDyn) partitions write requests depending on the least response times predicted by the SSD and HDD models. MixDyn maintains a table to store information about the current location of data (device id) and updates it whenever some data is migrated from one device to the other. Read requests are always serviced from the device which contains the data.

Figure 10(a) illustrates the performance of MixedStore incorporating the prediction models in MixDyn with respect to a disk-only and flash-only system for the random write dominant Financial trace. Although we observe good performance from flash device for servicing most requests, but some requests suffer from extensive GC overhead thus exhibiting high response time on flash. Our prediction model is able to move these requests to the disk and achieve better performance for MixedStore. Moreover, MixedStore reduces the average system response time by about 71% as compared to a disk-only system. Similar performance improvement is observed for Cello. However, the limitation of simplistic disk prediction model is observed for Cello in Figure 10(b) where flash-only system improves response time by about 20% as compared to MixedStore. The disk prediction model (in MixedStore) is unable to capture the high intensity of random writes resulting in incorrect prediction by MixDyn since some high latency requests are now inevitably wrongly serviced from disk. We believe with a more sophisticated disk performance prediction model will alleviate such discrepancies and improve the performance of MixDyn. We plan to pursue this as part of our future work.



(a) Log-scale



(b) Log-scale

Figure 10: Performance of MixedStore compared with a disk-only and a flash-only system. (a) and (b) show the CDF of system response time for Financial Trace and Cello99. *The 99% confidence intervals are very small and hence not shown.*

5.4.2 Adaptive Wear-Leveling

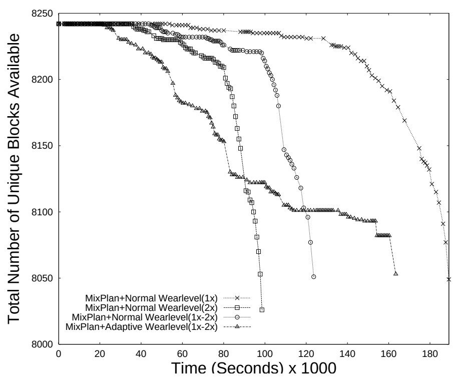


Figure 12: Adaptive Wear-Leveling. 1x and 2x denote the normal and unanticipated increase (2 times speed-up) in I/O intensity in the Financial trace. 1x-2x denotes a trace with regions of normal and increased I/O activity. Normal wear-leveling refers to continuously invoking wear-leveling algorithms irrespective of the available useful lifetime of blocks on flash.

The lifetime projections made by MixPlan are subject to violations due to uncertainties (increased unanticipated I/O activity) in the enterprise-scale workloads. Our novel adaptive wear-leveler helps MixDyn in upholding these guarantees. Figure 12 shows the impact of our adaptive wear-leveler on the Financial trace with modified inter-arrival times to resemble a workload with periods of high I/O activity. It halts the wear-leveling algorithm when it detects prolonged unanticipated I/O activity (we use static profiling to detect these periods). As compared to the normal wear-leveling algorithm which continuously performs leveling irrespective of the residual lifetime of blocks, our adaptive algorithm is able to improve the useful lifetime of flash by about 33%. This helps in delaying the need for replacement and reducing re-deployment costs. This enables MixDyn to achieve the lifetime guarantees as projected by MixPlan; hence both our capacity planning and dynamic-controller tools act in tandem to achieve the lifetime and performance budgetary requirements.

5.4.3 MixDyn at Work: A Microscopic Look

Figure 11 (next page) shows MixDyn at work for parts of the financial trace. Our dynamic data partitioner is able to intelligently partition the incoming requests, thus improving the system response time as compared to the static data partitioner. The fragmentation buster is able to reduce the tail of the CDF of response time (Figure 11-(b)), thus reducing the number of requests that experience high response time. We experiment with a write regulator that detects increased I/O activity and consistently monitors the expected flash life through the lifetime model of MixPlan.

We experiment with two models of a static write rate regulator that pick 25% or 50% (uniformly at random) of the requests being sent to flash and redirect them to HDD during periods of higher-than-expected I/O intensity. Let us call these policies Red_{25} *and* Red_{50}, *respectively.* Figure 11-(c) shows that we are able to reduce the flash block erase rate by about 25% while reducing the requests being serviced by flash by about 21% using Red_{25}. An additional 19% reduction in the erase rate is observed using Red_{50}. However, it results in an increase of 0.83ms in average system response time. Thus, the rate of write regulation must be chosen judiciously so as to meet the performance budget while ensuring that lifetime guarantees are satisfied. Thus, MixDyn acting in concert with MixPlan is able to judiciously utilize the different mechanisms to uphold the lifetime and performance requirements.

6. CONCLUDING REMARKS

We developed an on-line capacity planner called *MixPlan* that used statistical models to provide storage administrators with guidelines on provisioning a hybrid system in a cost-effective manner. We then developed a dynamic controller, *MixDyn*, that used shorter time-scale SSD and HDD models along with regulation of write-rate to the SSD and a novel idea of adaptive wear-leveling within the SSD to operate the storage system within regions of desirable cost, performance, and lifetime budgets. We evaluated these systems using MixedSim with a variety of well-regarded benchmarks. We found that MixedStore is able to reduce the average system response time by about 71% as compared to a HDD-based system for a enterprise-scale Financial trace. Moreover, our innovative adaptive wear-leveling mechanism was able to prolong the life of SSDs by about 33% in the presence of unanticipated increase in I/O intensity.

7. REFERENCES

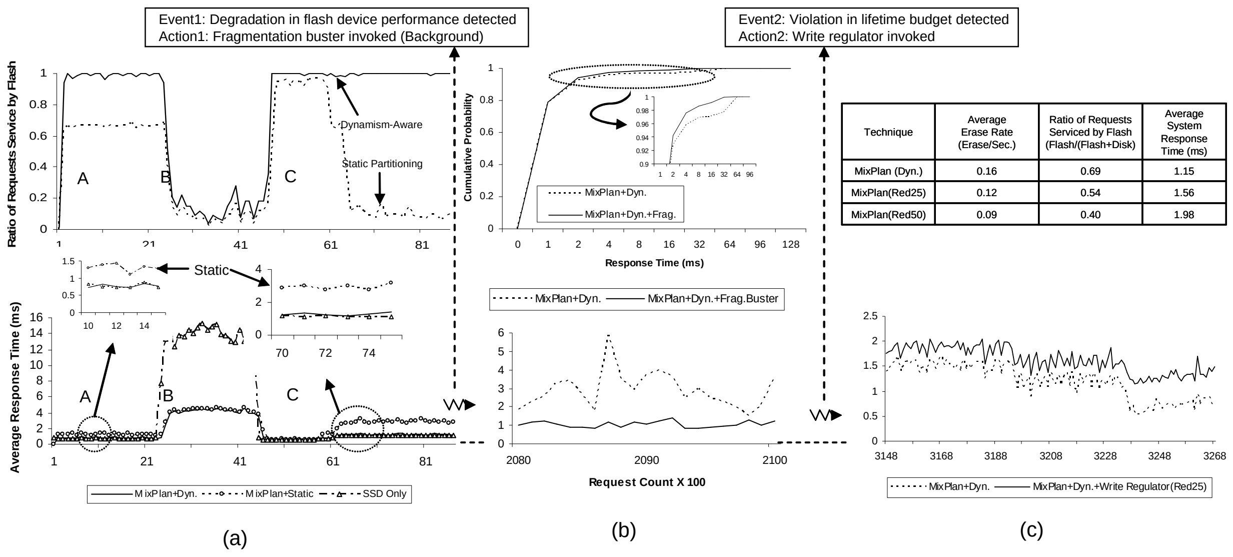
[1] Flash vs DRAM Price Projections . `http://www.storagesearch.com/ssd-ram-flash%20pricing.html`.

[2] Samsung 256GB Flash SSD With High-Speed Interface . `http://www.i4u.com/article17560.html`.

[3] 2.5" MTron SATA Solid State Drive - MSP 7000. `http://www.mtron.net/English/Product/ec_msp7000.asp`.

[4] 2.5" Super-Talent SATA Solid State Drive. `http://www.supertalent.com/products/ssd-commercial.php?type=SATA`.

[5] T. Bisson and S. A. Brandt. Reducing Hybrid Disk Write Latency with Flash-Backed I/O Requests. In *International Symposium on Modeling, Analysis,*



Technique	Average Erase Rate (Erase/Sec.)	Ratio of Requests Serviced by Flash (Flash/(Flash+Disk)	Average System Response Time (ms)
MixPlan (Dyn.)	0.16	0.69	1.15
MixPlan(Red25)	0.12	0.54	1.56
MixPlan(Red50)	0.09	0.40	1.98

Figure 11: MixDyn at work: The x-axis represents the progression of requests in Financial trace. Dyn., frag. buster and Write Regulator are various MixDyn mechanisms acting in concert with MixPlan. (a) compares the performance of MixPlan along with dynamism-aware data partioner(dyn.) with a static data partitioning policy and a SSD only system. Region B represents a period of intense, large, sequential write requests requiring GC on SSD and degrading the performance in a SSD only system. In regions A and C, the dynamism-aware data partitioner is able to make better performance predictions than a static policy, thus reducing the response time. (b) MixDyn observes degradation in SSD performance and invokes Fragmentation buster (Frag. Buster) during idle periods, thus providing sustained improved performance. (c) MixDyn observes violation in lifetime guarantees made by MixPlan and invokes the write regulator causing small degradation in response time since some requests which could have been serviced from flash are now sent to disk. However, this helps in reducing the erase rate and meeting lifetime budget while still meeting performance guarantees. Red_{25} and Red_{50} represent different write-regulation policies.

and Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE Computer Society, 2007.

[6] Y.-H. Chang, J.-W. Hsieh, and T.-W. Kuo. Endurance enhancement of flash-memory storage systems: An efficient static wear leveling design. In *Proceedings of the 44th Annual Conference on Design Automation*, pages 212–217, New York, NY, USA, 2007. ACM.

[7] S. Charrap, P. Lu, and Y. He. Thermal Stability of Recorded Information at High Densities. *IEEE Transactions on Magnetics*, 33(1):978–983, January 1997.

[8] J. Chen and J. Moon. Detection Signal-to-Noise Ratio versus Bit Cell Aspect Ratio at High Areal Densities. *IEEE Transactions on Magnetics*, 37(3):1157–1167, May 2001.

[9] T. Chung, D. Park, S. Park, D. Lee, S. Lee, and H. Song. System Software for Flash Memory: A Survey. In *Proceedings of the International Conference on Embedded and Ubiquitous Computing*, pages 394–404, August 2006.

[10] E. Gal and S. Toledo. Algorithms and Data Structures for Flash Memories. *ACM Computing Survey*, 37(2):138–163, 2005.

[11] Flash Price Drop Spurs Innovation, Feb 2008. `http://www.washingtonpost.com/wp-dyn/content/article/2008/02/01/AR2008020101313.html`.

[12] Can flash memory become the foundation for a new tier in the storage hierarchy?, Sept 2008. `http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=547`.

[13] G. Ganger, B. Worthington, and Y. Patt. *The DiskSim Simulation Environment Version 3.0 Reference Manual*.

[14] P. Goyal, D. Modha, and R. Tewari. CacheCOW: Providing QoS for Storage System Caches. *SIGMETRICS Performance Evaluation Review*, 31(1):306–307, 2003.

[15] A. Gulati, A. Merchant, and P. J. Varman. pClock: An Arrival Curve based Approach for QoS Guarantees in Shared Storage Systems. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 13–24, June 2007.

[16] A. Gupta, Y. Kim, and B. Urgaonkar. DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating System (To Appear)*, 2009.

[17] S. Gurumurthi, A. Sivasubramaniam, and V. Natarajan. Disk Drive Roadmap from the Thermal Perspective: A Case for Dynamic Thermal Management. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 38–49, June 2005.

[18] HP Labs. Tools and Traces. `http://tesla.hpl.hp.com/public_software/`.

[19] Intel, STMicroelectronics Deliver Industry's First Phase Change Memory Prototypes. `http://www.intel.com/pressroom/archive/releases/20080206corp.htm`.

[20] J. Kim, J.M. Kim, S.H. Noh, S. Min, and Y. Cho. A Space-Efficient Flash Translation Layer for Compactflash Systems. *IEEE Transactions on Consumer Electronics*, 48(2):366–375, 2002.

[21] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley-Interscience, 1991.

[22] JFFS2: The Journalling Flash FileSystem. `http://sources.redhat.com/jffs2/jffs2.pdf`.

[23] D. Jung, Y. Chae, H. Jo, J. Kim, and J. Lee. A Group-based Wear-Leveling Algorithm for Large-Capacity Flash Memory Storage Systems. In *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, pages 160–164, September 2007.

[24] J. Kang, H. Jo, J. Kim, and J. Lee. A Superblock-based Flash Translation Layer for NAND Flash Memory. In *Proceedings of the International Conference on Embedded Software (EMSOFT)*, pages 161–170, October 2006.

[25] A. Kawaguchi, S. Nishioka, and H. Motoda. A Flash-Memory based File System. In *Proceedings of the Winter 1995 USENIX Technical Conference*, pages 155–164, 1995.

[26] T. Kgil, D. Roberts, and T. Mudge. Improving NAND Flash Based Disk Caches. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, June 2008.

[27] H. Kim and S. Ahn. BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, pages 1–14, Feburary 2008.

[28] Y. Kim, S. Gurumurthi, and A. Sivasubramaniam. Understanding the Performance-Temperature Interactions in Disk I/O of Server Workloads. In *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*, Febuary 2006.

[29] S. Lee and B. Moon. Design of Flash-based DBMS: An In-Page Logging Approach. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 55–66, August 2007.

[30] S. Lee, D. Park, T. Chung, D. Lee, S. Park, and H. Song. A Log Buffer based

Flash Translation Layer Using Fully Associative Sector Translation. *IEEE Transactions on Embedded Computing Systems*, 6(3):18, 2007.

[31] S. Lee, D. Shin, Y. Kim, and J. Kim. LAST: Locality-Aware Sector Translation for NAND Flash Memory-Based Storage Systems. In *Proceedings of the International Workshop on Storage and I/O Virtualization, Performance, Energy, Evaluation and Dependability (SPEED2008)*, Feburary 2008.

[32] A. Leventhal. Flash Storage Memory. *Communications of the ACM*, 51(7):47–51, 2008.

[33] K. M. J. Lofgren, R. D. Norman, G. B. Thelin, and A. Gupta. Wear Leveling Techniques for Flash EEPROM. In *United States Patent, No 6,850,443*, 2005.

[34] M. Mallary, A. Torabi, and M. Benakli. One Terabit Per Square Inch Perpendicular Recording Conceptual Design. *IEEE Transactions on Magnetics*, 38(4):1719–1724, July 2002.

[35] Managed Flash Technology (MFT) Flash SSD Acceleration Software. `http://www.easyco.com/zx1285301141358249458/mft/index.htm`.

[36] B. Marsh, F. Douglis, and P. Krishnan. Flash memory file caching for mobile computers. In *To appare in Proceedings of the 27th Hawaii Conference on Systems Science*, 1994.

[37] N. Mi, A. Riska, E. Smirni, and E. Riedel. Enhancing Data Availability through Background Activities. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 492–501, 2008.

[38] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating enterprise storage to ssds: Analysis of tradeoffs. In *Microsoft Research Technical Report MSR-TR-2008-169*, November 2008.

[39] H. Niijima. Design of a Solid-State File Using Flash EEPROM. *IBM Journal of Research and Developement*, 39(5):531–545, 1995.

[40] A. Nitin, P. Vijayan, and W. Ted. Design Tradeoffs for SSD Performance. In *Proceedings of the USENIX Annual Technical Conference*, June 2008.

[41] OLTP Trace from UMass Trace Repository. `http://traces.cs.umass.edu/index.php/Storage/Storage`.

[42] V. Prabhakaran, T. L. Rodeheffer, and L. Zhou. Transactional Flash . In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI) (To Appear)*, December 2008.

[43] S. Tehrani and J. Slaughter and E. Chen and M. Durlam and J. Shi and M. DeHerren. Progress and Outlook for MRAM Technology. *IEEE Transactions on Magnetics*, 35(5):2814–2819, September 1999.

[44] N. Schirle and D. Lieu. History and Trends in the Development of Motorized Spindles for Hard Disk Drives. *IEEE Transactions on Magnetics*, 32(3):1703–1708, May 1996.

[45] B. Schroeder and G. A. Gibson. Understanding Disk Failure Rates: What does an MTTF of 1,000,000 hours mean to you? In *Proceedings of the Annual Conference on File and Storage Technology (FAST)*, Feburary 2007.

[46] Y. Shimada. FeRAM: Next Generation Challenges and Future Directions. *Electronics Weekly*, May 2008.

[47] M. Uysal, A. Merchant, and G. A. Alvarez. Using MEMS-Based Storage in Disk Arrays. In *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 89–101, Berkeley, CA, USA, 2003. USENIX Association.

[48] White Paper: Datacenter SSDs: Solid Footing for Growth. `http://www.samsung.com/global/business/semiconductor/products/flash/FlashApplicationNote.html`.

[49] M. Wu and W. Zwaenepoel. eNVy: a Non-Volatile Main Memory Storage System. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 86–97, 1994.

[50] YAFFS: Yet Another Flash File System. `http://www.aleph1.co.uk/yaffs`.

[51] ZFS L2ARC. `http://opensolaris.org/os/community/arc/caselog/2007/618/`.

[52] J. Zhang, A. Sivasubramaniam, H. Franke, N. Gautam, Y. Zhang, and S. Nagar. Synthesizing Representative I/O Workloads for TPC-H. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, 2004.